AF
IFW

Our Docket No.: 1496.00047

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicant	Ariel Cohen et al.		
Application No.:	09/746,796	Examiner:	Meonske, T.
Filed:	December 22, 2000	Art Group:	2183
For:	MICROCODE BASED HARDWARE TRANSLATOR TO SUPPORT MULTITUDE OF PROCESSORS		

I hereby certify that this letter, the response or amendment attached hereto are being deposited with the United States Postal Service as first class mail in an envelope addressed to Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on July 11, 2005.

By: Jan M. Dunbar
Jan M. Dunbar

REPLY BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Appellants submit the following Reply Brief pursuant to 37 C.F.R. §41.41 for consideration by the Board of Patent Appeals and Interferences.

TABLE OF CONTENTS

I. STATUS OF CLAIMS

Claims 1-22 are pending and remain rejected. The Appellants hereby maintain the appeal of the rejections of claims 1-22.

II. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-6, 8, 9, 12-14, 17-19 and 22 are rejected under 35 U.S.C. §102(b) as being anticipated by Hilgendorf et al.[1]

Claims 10, 11 and 20 are rejected under 35 U.S.C. §103(a) as being unpatentable over Hilgendorf in view of Martin.[2]

Claim 7 is rejected under 35 U.S.C. §103(a) as being unpatentable over Hilgendorf.

Claim 15 is rejected under 35 U.S.C. §103(a) as being unpatentable over Hilgendorf with Gee et al.,[3] Wilmot, II,[4] Favor[5] and Aravamudan et al.[6] cited for extrinsic evidence.

Claims 16 and 21 are rejected under 35 U.S.C. §103(a) as being unpatentable over Hilgendorf in view of Gee et al. with Wilmot, Favor and Aravamudan cited for extrinsic evidence.

---

[1] U.S. Patent No. 5,925,124 (hereinafter Hilgendorf).

[2] U.S. Patent No. 4,439,828.

[3] U.S. Patent No. 6,374,286 (hereinafter Gee).

[4] U.S. Patent No. 6,615,340 (hereinafter Wilmot).

[5] U.S. Patent No. 5,926,642.

[6] U.S. Patent No. 6,502,109 (hereinafter Aravamudan).

III. ARGUMENTS IN RESPONSE TO EXAMINER'S ANSWER

A. Rejections under 35 U.S.C. § 102.

As set forth on pages 2-11 of the Examiner's Answer,[7] claims 1-6, 8, 9, 12-14, 17-19 and 22 are rejected under 35 U.S.C. § 102(b) as being anticipated by Hilgendorf et al.[8]

The Federal Circuit has stated that "[t]o anticipate, *every element and limitation* of the claimed invention must be found in a single prior art reference, *arranged as in the claim.*"[9] The Federal circuit has added that the anticipation determination is viewed from one of ordinary skill in the art: "There must be no difference between the claimed invention and the reference disclosure, as viewed by a person of ordinary skill in the field of the invention."[10] Furthermore, "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference."[11] As explained herein below, because Hilgendorf does not disclose or suggest a microcode memory **containing sequences of instruction codes of a second instruction set that emulate a functionality of the instruction codes of the first instruction set**, Hilgendorf does not anticipate the presently claimed invention.

---

[7] Dated May 19 2005.

[8] U.S. Patent No. 5,925,124 (hereinafter Hilgendorf).

[9] *Brown v. 3M*, 60 USPQ2d 1375, 1376 (Fed. Cir. 2001) citing *Karsten Mfg. Corp. v. Cleveland Golf Co.*, 242 F.3d 1376, 1383, 58 USPQ2d 1286, 1291 (Fed. Cir. 2001); *Scripps Clinic & Research Found. v. Genentech Inc.*, 927 F.2d 1565, 18 U.S.P.Q.2d 1001, 1010 (Fed. Cir. 1991) (Emphasis added by Appellant).

[10] *Scripps Clinic & Research Found. v. Genentech Inc.*, 927 F.2d 1565, 18 U.S.P.Q.2d 1001, 1010 (Fed. Cir. 1991).

[11] *Verdegaal Bros. V. Union Oil Co. of California*, 814 F.2d 628, USPQ2d 1051, 1053 (Fed Circ. 1987).

1. Claims 1-3, 5-9, 12-14, 17-19 and 22 are fully patentable over Hilgendorf.

The presently pending claim 1 provides an apparatus comprising a circuit configured to translate instruction codes of a first instruction set on-the-fly into addresses into a microcode memory **containing sequences of instruction codes of a second instruction set that emulate a functionality of the instruction codes of the first instruction set**. Claims 17 and 18 include similar limitations. Claims 2, 3, 5-9, 12-14, 19 and 22 depend, directly or indirectly, from either claim 1 or claim 19.

The Examiner's interpretation of the phrase "instruction codes of a second instruction set" as referring to "opcodes" is not reasonable in light of the present specification. Furthermore, the Examiner does not explain why an extrinsic source (i.e., Microsoft Computer Dictionary, 4th Ed.) is used instead of the specification to interpret the claim language.[12] Contrary to the position taken by the Examiner, one of ordinary skill in the art would understand the phrase "sequences of instruction codes of a second instruction set" as referring to "instructions of the second instruction set." Specifically, the specification states:

> In response to an instruction address having a predetermined value (e.g., an intercept program address or IPA), the circuit 100 may be configured to translate instruction codes of a first instruction set received from the memory system 104 into **a sequence of instruction codes of a second instruction set (e.g., native instructions) for the CPU 102.** The circuit 100 may be configured to generate instruction addresses that may be presented to the memory system 104 in place of the instruction addresses generated by the CPU 102. In one example, the circuit 100 may be used to translate **instructions from a first instruction set (e.g., instruction codes of the programming language Java or any other programming languages or instruction sets) into sequences of the native instruction codes for**

12

> **the CPU 102**. The sequences of native instruction codes will generally emulate the functionality of the instruction codes of the first instruction set. The circuit 100 may be configured to operate invisibly to the CPU 102 and the memory system 104.[13]

Thus , the terms "instructions" and "instruction codes" are used interchangeably in the specification.

Further examples of this usage can be found in the specification. For example, on page 27, lines 3-11 the specification states:

> The circuit 100 generally fetches bytecodes from the memory 104. The bytecodes may include both instructions as well as data. The circuit 100 generally buffers the bytecode and then decodes the bytecode. The decoded bytecode may generate an address to the microcode **where the corresponding native instructions stream (NIS) may be located**. The decoded bytecode may include virtual stack references that may be resolved by the configuration registers 200 and/or the stack management unit 204 according to the stack status.[14]

On page 30, lines12-15 the specification states:

> The microcode memory 210 may have an output that may present a signal to the native instruction generator (NIG) block 212 and an input that may receive a signal from the sequence optimization unit (SOU) 218. **The microcode memory 210 may comprise the NIS [native instructions stream] for each bytecode.** The microcode memory 210 may comprise a number of virtual stack register names and an indication of the bit fields that the NIG 212 may replace with the actual registers codes. By using a memory based architecture for the translation of the bytecodes and using a pre-defined format, the microcode memory 210 and the NIG 212 may be easily replaced to match any kind of processor 102. The flexibility of the circuit 100 may be affected by the design of the microcode memory 210. The microcode memory 210 may also hold a memory pointer and down counter. The memory pointer may be incremented with each microcode read and the down-counter may be decremented. The

---

[13] See page 11, line 15 through page 12, line 10 of the specification, emphasis added.

[14] Page 27, lines 3-11 of the specification, emphasis added.

> pointer may first be loaded with an NIS address generated by the Decoder/LUT block 208. The counter may first be loaded with the number of instructions within the current NIS. The microcode memory 210 may receive addresses from the Decoder/LUT 208 of the requested NIS, addresses from the sequence optimizing unit 218, and control signals from the control unit 216 to select which addresses to use.[15]

Another examples can be found on page 30 where the specification states:

> The microcode generally does not contain a simple set of native instructions for the processor to execute. Rather, **the microcode may contain instructions** such as: add TOS,TOS-1,TOS-1, where TOS and TOS-n are actually virtual stack registers.[16]

Furthermore, one skilled in the art would understand an instruction to be "an action statement in any computer language, most often in machine or assembly language."[17] One skilled in the art would further understand an opcode to be "the **portion of** a machine language or assembly language **instruction** that specifies the type of instruction and the structure of the data on which [the instruction] operates."[18] The understanding that an opcode is not an instruction, but rather only a part of an instruction is also evidenced by Hilgendorf. Specifically, Hilgendorf states :

> The external instruction 100, which is **an instruction** of code A, **comprises** an OP-code of code A (102), an "immediate" data field (110), logical register identifiers (111, 112) which specify the instruction's source and target registers, and additional information

---

[15] Page 22, line 3 through page 23, line 2 of the specification, emphasis added.

[16] Page 30, lines 11-15 of the specification, emphasis added.

[17] See definition of "instruction" in Microsoft Computer Dictionary, 5th Ed., Microsoft Corporation, 2002, pg. 276.

[18] See definition of "operation code" in Microsoft Computer Dictionary, 5th Ed., Microsoft Corporation, 2002, pg. 378.

about the instruction, such as status bits.[19]

Hilgendorf further states:

> Rearrangement information needed for converting the external instruction (100) to a set of internal instructions (101) is contained in a translation table (106). Said translation table can either be implemented as a ROM table or as a RAM table. Each entry (105) of said table corresponds to one of said external code A instructions (100), and determines how the instruction elements of this specific instruction are to be rearranged **in order to form the corresponding internal instructions** (101).[20]

Thus, Hilgendorf clearly teaches that the internal instructions of code B (101) are NOT in the translation table 106, but **are formed externally to the translation table 106** using rearrangement information from the translation table 106.[21]

Therefore, assuming, *arguendo*, the translation table 106 of Hilgendorf would be considered by one of ordinary skill in the art to be similar to the presently claimed microcode memory,[22] because (i) the translation table 106 of Hilgendorf contains only opcodes of code B instead of the entire instructions of code B and (ii) Hilgendorf teaches that the internal instructions of code B are formed externally to the translation table 106, Hilgendorf does not disclose or suggest a microcode memory **containing sequences of instruction codes of a second instruction set that emulate a functionality of the instruction codes of the first instruction set**, as presently claimed.

---

[19] Column 6, line 62-67 of Hilgendorf, emphasis added.

[20] Column 7, lines 9-17 of Hilgendorf, emphasis added.

[21] See FIGS. 1 and 5 and column 7, lines 9-34 of Hilgendorf.

[22] As suggested by the Examiner in section 5 on page 2 of the final Office Action dated May 18, 2004 and for which Appellant's representative does not necessarily agree.

Therefore, Hilgendorf does not disclose or suggest each and every element of the presently claimed invention, arranged as in the claims, as required by MPEP § 2131. As such, the presently claimed invention is fully patentable under 35 U.S.C. § 102 over the cited reference and the rejection should be reversed.

2. Claim 4 is fully patentable over Hilgendorf.

The presently pending claim 4 depends directly from claim 1 and, therefore, includes all the limitations of claim 1. Consequently, the arguments presented above in support of claim 1 are incorporated herein by reference in support of claim 4. Claim 4 further recites that predetermined sequences of the instruction codes of the first instruction set are used to address the microcode memory. Hilgendorf does not disclose or suggest that predetermined sequences of the instruction codes of the first instruction set are used to address the microcode memory, as presently claimed.

Specifically, Hilgendorf states:

> Rearrangement information needed for converting the external instruction (100) to a set of internal instructions (101) is contained in a translation table (106). Said translation table can either be implemented as a ROM table or as a RAM table. **Each entry (105) of said table corresponds to ONE of said external code A instructions** (100), and determines how the instruction elements of this specific instruction are to be rearranged in order to form the corresponding internal instructions (101).
>
> As **EACH translation table entry (105) corresponds to ONE external code A instruction (100), the OP-code (102) of said external instruction can be used to determine the correct translation table entry**. This is done by forwarding the OP-code (102) to an address generation logic (104), which converts said OP-code to the address of the corresponding entry in the table. This address is then used to access (103) the corresponding translation table entry (105). Said entry contains the OP-codes of all the internal

> code B instructions (107) to which the external instruction is to be converted to, and multiplexer control information (129), which is used for controlling the recombination of the external instruction's elements.[23]

Since each translation table entry corresponds to ONE external code A instruction, it follows that Hilgendorf does not disclose or suggest that **predetermined sequences of the instruction codes of the first instruction set** are used to address the microcode memory, as presently claimed. Thus, Hilgendorf does not disclose or suggest each and every element of the presently claimed invention, arranged as in the present claims. Therefore, the Examiner has failed to meet the Office's burden under MPEP §2131 to factually establish a *prime facie* case of anticipation. As such, claim 4 is fully patentable over the cited reference and the rejection should be reversed.

Furthermore, The Examiner argues language which is not contained in the claim. Specifically, the Examiner states:

> The translation table is designed to have any and all predetermined sequences, or programs, of the First Instruction set address the table for instruction translation. A program for a computer system that for all of eternity only performs one single instruction is not useful. A program for a computer system must have a sequence of several instructions. Any sequence of instructions contains a plurality of subsequences. So a program necessarily contains several sequences. Therefore Hilgendorf has in fact taught predetermined sequences of said instruction codes of said first instruction set (a plurality of subsequences in a program) are used to address said microcode memory.[24]

The presently pending claim 4 does not recite a plurality of subsequences of a program as argued by the Examiner. Since the Examiner is arguing language which is not even contained in the claim, the

---

[23] Column 7, lines 9-30 of Hilgendorf, emphasis added.

[24] See page 5, lines 1-6 of the Examiner's Answer dated May 9, 2005.

Examiner's arguments are moot.

Furthermore, the fact that the Examiner must resort to limitations that are neither recited in the claim nor recited in the reference (i.e., a plurality of subsequences in a program) further evidences that Hilgendorf does not disclose or suggest each and every element of the claimed invention, arranged as in the claims, as required by MPEP § 2131. Hilgendorf explicitly states that EACH translation table entry corresponds to ONE external code A instruction.

In contrast to Hilgendorf, the specification provides an example of using **predetermined sequences of the instruction codes of the first instruction set** to address the microcode memory. In particular, the specification states:

> The Sequence Optimization Unit (SOU) 218 may be configured to look for any of a plurality of pre-defined sequences of bytecodes. **When a sequence is detected, the SOU 218 may notify the controller 216 and take over control of the address pointer into the microcode memory 210.** By doing so, the SOU 218 may facilitate the generation of an optimized native instruction sequence to the processor 102. **An example may be illustrated by the following sequence of bytecodes: load immediate value to stack, add the immediate value to the value preceding it in the stack, and put the result back in the stack. In some processors that support an immediate mode, the example sequence may be executed in a single CPU instruction such as add r5,#45,r5. Thus, instead of generating two instructions (e.g., mov #5,r6 and add r5,r6,r5), an optimized single instruction may be generated.** The generation of optimized code may enable further speed increase and faster execution of the translated instruction set (e.g., the Java code). In general, each processor (e.g., MIPS, ARM, 68000, etc.) may have a unique instruction set. An optimization for one processor may not be possible for another and vice-versa. The SOU 216 may be implemented, in one example, with microcode to support optimization for any possible processor.[25]

---

[25] See page 26, line 3 through page 27, line 2 of the specification, emphasis added.

A person of ordinary skill in the art would not view a translation table where EACH translation table entry corresponds to ONE external code A instruction to be the same as a microcode memory that can be addressed using **predetermined sequences of the instruction codes of the first instruction set**, as presently claimed. Therefore, the Examiner has not met the Office's burden to factually support a *prima facie* conclusion that Hilgendorf discloses or suggests each and every element of the claimed invention, arranged as in the claims, as required by MPEP § 2131. As such, presently pending claim 4 is fully patentable under 35 U.S.C. §102 over Hilgendorf and the rejection should be reversed.

B. Rejections under 35 U.S.C. § 103

As set forth on pages 7-9 of the Examiner's Answer,[26] claims 7, 10, 11, 15, 16, 20 and 21 are rejected under 35 U.S.C. §103(a) as being unpatentable over either Hilgendorf alone (claims 7 and 15), Hilgendorf in combination with Martin (claims 10, 11 and 20) or Hilgendorf in combination with Gee (claim 16 and claim 21).

The Examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness.[27] If the Examiner does not produce a *prima facie* case, the Applicant is under no obligation to submit evidence of nonobviousness.[28] "[T]o establish obviousness based on a combination of the elements disclosed in the prior art, there must be some motivation, suggestion

---

[26] Dated May 9, 2005.

[27] Manual of Patent Examining Procedure (M.P.E.P.), Eighth Edition, Rev. 2, May 2004, §2142.

[28] *Id.*

or teaching of the desirability of making the specific combination that was made by the applicants."[29] "[T]he factual inquiry whether to combine references must be thorough and searching."[30] "This factual question ... [cannot] be resolved on subjective belief and unknown authority."[31] "It must be based on objective evidence of record."[32] The Examiner must show that (a) there is some suggestion or motivation, either in the references or in the knowledge generally available to one of ordinary skill in the art, to modify or combine the references, (b) there is a reasonable expectation of success, and (c) the prior art reference (or combination of references) teaches or suggests all of the claim limitations.[33]

The Federal Circuit has held that both the suggestion to modify or combine the references and the reasonable expectation of success must be found in the prior art itself, not merely in Appellant's disclosure.[34] Furthermore, the Court of Appeals for the Federal Circuit has indicated that the requirement for showing the teaching of motivation to combine references is "rigorous" and must be "clear and particular."[35] Furthermore, the Board has held that the claimed invention is

---

[29] *In re Kotzab*, 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1316 (Fed. Cir. 2000) (citing *In re Dance*, 160 F.3d 1339, 1343, 48 USPQ2d 1635, 1637 (Fed. Cir. 1998); *In re Gordon*, 733 F.2d 900, 902, 221 USPQ 1125, 1127 (Fed. Cir. 1984)).

[30] *McGinley v. Franklin Sports, Inc.*, 262 F.3d 1339, 1351-52, 60 USPQ2d 1001, 1008 (Fed. Cir. 2001).

[31] *In re Lee*, 277 F.3d 1338, 1343-44, 61 USPQ2d 1430, 1434 (Fed. Cir. 2002).

[32] *Id.* at 1343, 61 USPQ2d at 1434.

[33] M.P.E.P. §2142.

[34] See *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438, 1442 (Fed. Cir. 1991).

[35] *In re Anita Dembiczak* and *Benson Zinbarg*, 50 U.S.P.Q.2d 1614 (Fed. Cir. 1999).

obvious only if either the references expressly or implicitly suggest the claimed invention, or **a convincing line of reasoning is presented by the examiner as to why an artisan would have found the claimed invention to be obvious in light of the teachings of the cited references.**[36]

1. Claims 10, 11 and 20 are fully patentable over Hilgendorf and Martin.

Claim 10 depends directly from claim 1 and, therefore, includes all of the limitations of claim 1. Consequently, the arguments presented above in support of claim 1 are incorporated herein by reference in support of claim 10. Claim 10 further provides that the circuit is configured to detect optimizable sequences of instruction codes on-the-fly. Claim 20 includes a similar limitation. Claim 11 depends directly from claim 10.

The Examiner admits that with respect to claims 10, 11 and 20 Hilgendorf has not taught wherein the circuit is configured to detect optimizable sequences of instruction codes on-the-fly.[37] Hilgendorf teaches an apparatus and a method for converting instructions of a code A to instructions of a code B by rearranging elements of a single instruction of code A to form one or more instructions of code B.[38] The conversion process of Hilgendorf is performed on single instructions of code A.[39] The modification proposed by the Examiner would change the principle of operation of Hilgendorf. If the proposed modification or combination of the prior art would

---

[36] See *Ex Parte Clapp*, 227 U.S.P.Q. 972, 973 (Bd. Pat. App. & Inter. 1985) (emphasis added by Appellant).

[37] See section 21 on page 7 of the final Office Action dated May 18, 2004.

[38] See abstract and column 7, lines 9-30 of Hilgendorf.

[39] *Id.*

change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious.[40] Furthermore, in contrast to the Examiner's position that Hilgendorf's suggestion to exploit parallelism in superscaler processing is motivation for detecting optimizable sequences,[41] Hilgendorf states that the parallelism can be exploited by processing instructions out of their sequential order using various execution units.[42] Since Hilgendorf teaches the parallelism is exploited by processing instructions out of sequential order, it follows that Hilgendorf's reference to exploiting parallelism does not appear to teach or suggest detecting optimizable sequences of instruction codes on-the-fly, as presently claimed. Therefore, the Examiner failed to meet the Office's burden to factually establish a *prima facie* conclusion of obviousness.[43] As such, claims 10, 11 and 20 are fully patentable over the cited references and the rejection should be reversed.

2. Claim 15 is fully patentable over Hilgendorf

Claim 15 depends directly from claim 1 and, therefore, includes all of the limitations of claim 1. Consequently, the arguments presented above in support of claim 1 are incorporated herein by reference in support of claim 15. Claim 15 further provides that the instruction codes of the first instruction set comprise Java bytecodes.

---

[40] See MPEP §2143.02 citing *In re Ratti*, 270 F.2d 810, 123 USPQ 349 (CCPA 1959).

[41] See section 20 on pages 11-12 of the Examiner's Answer dated May 9, 2005.

[42] See column 1, lines 56-62 of Hilgendorf.

[43] M.P.E.P. §2142.

The Examiner admits that with respect to claim 15 Hilgendorf has not explicitly taught wherein said instruction codes of said first instruction set comprise Java bytecodes.[44] The Examiner does not adequately address where in the prior art references the reasonable expectation of success can be found for applying the conversion of Hilgendorf to Java bytecodes. The Federal Circuit has held that both the suggestion to modify or combine the references and the reasonable expectation of success must be found in the prior art itself, not merely in Appellant's disclosure.[45] Hilgendorf is silent regarding Java bytecodes. Hilgendorf states that the external instruction 100 can be either a CISC instruction or a RISC instruction.[46] One of ordinary skill in the art would recognize that CISC instructions and RISC instructions are different from Java bytecodes. In particular, one of ordinary skill in the art would recognize that CISC instructions and RISC instructions can be directly executed by appropriate microprocessor designs.[47] In contrast, one of ordinary skill in the art would recognize Java bytecodes as an abstract, processor-independent form that cannot be directly executed by most CPUs.[48]

Hilgendorf teaches rearranging elements of an external CISC or RISC instruction to generate an internal instruction. The Examiner fails to present any evidence or convincing line of

---

[44] See section 24 on page 8 of the final Office Action dated May 18, 2004.

[45] See *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438, 1442 (Fed. Cir. 1991).

[46] Column 6, lines 54-57 of Hilgendorf.

[47] See definition of CISC and RISC in Microsoft Computer Dictionary, 5th Ed., Microsoft Corporation, 2002, pp. 100 and 455.

[48] See definition of bytecode in Microsoft Computer Dictionary, 5th Ed., Microsoft Corporation, 2002, pg. 79.

reasoning to factually support a conclusion that Java bytecodes have elements that can merely be rearranged using the apparatus of Hilgendorf to obtain internal instructions (i.e., instructions that can be directly executed by a processor). The position taken by the Examiner that it would have been obvious to one of ordinary skill in the art at the time of the invention to translate Java code into the host's computer code since Java is a popular and widely used programming language, and many programs would already be written in Java"[49] does not adequately address the issue of the reasonable expectation of success in using the apparatus or method of Hilgendorf.

Thus, because the Examiner failed to make the required showing that there is a reasonable expectation of success in using the invention of Hilgendorf with Java bytecodes, the Examiner failed to factually establish a *prima facie* conclusion of obviousness.[50] As such, the presently pending claim 15 is fully patentable over the cited references and the rejection should be reversed.

3. Claim 16 is fully patentable over Hilgendorf and Gee

Claim 16 depends directly from claim 1 and, therefore, includes all of the limitations of claim 1. Consequently, the arguments presented above in support of claim 1 are incorporated herein by reference in support of claim 16. Claim 16 further provides that the circuit comprises a portion of a Java virtual machine implemented in hardware.

The Examiner admits that Hilgendorf has not taught either a circuit comprising a

---

[49] See lines 10-14 on page 14 of the Examiner's Answer dated May 9, 2005.

[50] M.P.E.P. §2142.

portion of a Java virtual machine or a Java virtual machine implemented in hardware.[51] Gee does not cure the deficiencies of Hilgendorf. Gee discloses that Java bytecodes can be executed using either a Java run time environment called a Java virtual machine (JVM) or a full hardware direct execution Java processor (or Java machine).[52] Thus Gee teaches all software or full hardware implementations of a JVM. Therefore, Gee does not teach or suggest implementing only a portion of the Java virtual machine in hardware, as presently claimed. Thus, Gee does not teach or suggest each and every element of the presently claimed invention. Furthermore, the specification states that implementing **a portion** of the Java virtual machine in hardware provides advantages over a software only JVM and a full hardware implementation of a JVM.[53] Specifically, the specification states:

> The third technique involves a full hardware implementation of the JVM (i.e., HW-JVM or a picoJava core). The full hardware implementation of the JVM can provide a very fast executing machine. However, in order to use the full hardware implementation of the JVM, legacy code must be completely rewritten in Java. Therefore, the third technique is not backward compatible with older machines.[54]

The specification further states:

> In a preferred embodiment, the present invention may offer a much better performance than any software only based solution, with minimal memory footprint. In addition, the present invention may maintain compatibility to past designs and preserves the ability to run legacy code. The latter may provide a major advantage over

---

[51] Page 14, lines 20-21 and page 16, line 9 of the Examiner's Answer dated May 9, 2005.

[52] See column 2, lines 38-58 of Gee.

[53] See page 4, lines 7-13 and page 7, line 5 through page 8, line 5 of the specification.

[54] See page 4, lines 7-13 of the specification.

> conventional solutions since large investments made in the legacy code may be preserved.[55]

Therefore, the Examiner failed to meet the Office's burden of factually establishing a *prima facie* conclusion of obviousness.[56] As such, the presently pending claim 16 is fully patentable over the cited reference and the rejection should be reversed.

4. Claim 21 is fully patentable over Hilgendorf and Gee

Claim 21 depends directly from claim 1 and, therefore, includes all of the limitations of claim 1. Consequently, the arguments presented above in support of claim 1 are incorporated herein by reference in support of claim 21. Claim 21 further provides that the sequences of instruction codes of the second instruction set comprise one or more virtual stack references.

The Examiner admits that Hilgendorf has not taught wherein a sequence of instruction codes of an instruction set comprise one or more virtual stack references.[57] The Examiner fails to present any objective evidence that a sequence of instruction codes of an instruction set created by translating Java code with a Java virtual machine, as taught by Gee, would **necessarily** comprise one or more **virtual** stack references.[58] The text of Gee cited by the Examiner in support of the Examiner's position reads:

---

[55] See page 7, lines 11-18 of the specification.

[56] M.P.E.P. §2142.

[57] Page 16, lines 17-18 of the Examiner's Answer dated May 9, 2005.

[58] See page 18, lines 5-8 of the Examiner's Answer dated May 9, 2005 and page 9, section 27, lines 2-4 in the final Office Action dated May 18, 2004.

> The JVM is actually composed of one or more threads. Each JVM thread has a private JAVA stack, created at the same time as the thread, which stores JVM frames. A JAVA stack is the equivalent of [the] stack of a conventional programming language such as C. The JAVA stack holds local variables and partial results, and plays a part in method invocation and return. The JVM specification permits JAVA stacks to be of either a fixed or a dynamically varying size. The JVM also has a method area that is shared among all threads (column 7, lines 1-10 of Gee).

Nowhere in the above text does Gee expressly mention **virtual** stack references, as presently claimed. In contrast to the presently claimed virtual stack references, the JVM of Gee uses **actual** stack operations. Furthermore, the Examiner failed to present any objective evidence or convincing line of reasoning to support the conclusory statement that "in order to implement Java with a Java virtual machine it is **necessary** to have **virtual** stack references."[59] Therefore, the Examiner failed to meet the Office's burden of factually establishing a *prima facie* conclusion of obviousness by showing that each and every element of claim 21 is taught or suggested by the cited references.[60] As such, the presently pending claim 21 is fully patentable over the cited references and the rejection should be reversed.

---

[59] See page 18, section 30 of the Examiner's Answer dated May 9, 2005.

[60] M.P.E.P. §2142.

C. CONCLUSION

The cited references do not disclose or suggest a microcode memory containing sequences of instruction codes of a second instruction set that emulate a functionality of the instruction codes of the first instruction set, as recited in claims 1, 17 and 18. Hence, the Examiner has clearly erred with respect to the patentability of the claimed invention. It is respectfully requested that the Board overturn the Examiner's rejection of all pending claims, and hold that the claims are not rendered anticipated or obvious by the cited references. However, should the Board find the arguments herein in support of independent claims 1, 17 and/or 18 unpersuasive, the Board is respectfully requested to carefully consider the arguments set forth above in support of dependent claims 4, 10, 11, 15, 16, 20 and 21.

Respectfully submitted,

CHRISTOPHER P. MAIORANA, P.C.

Christopher P. Maiorana
Registration No. 42,829

Dated: July 11, 2005

c/o Pete Scott
Intellectual Property Law Department
LSI Logic Corporation
1621 Barber Lane
MS: D-106 Legal
Milpitas, CA 95035

VIII. CLAIM APPENDIX

The claims of the present application which are involved in this appeal are as follows:

1. An apparatus comprising:

a circuit configured to translate instruction codes of a first instruction set on-the-fly into addresses into a microcode memory containing sequences of instruction codes of a second instruction set that emulate a functionality of the instruction codes of the first instruction set.

2. The apparatus according to claim 1, wherein said sequences of instruction codes of said second instruction set generated in response to said instruction codes of said first instruction set are stored in a cache.

3. The apparatus according to claim 1, wherein said circuit comprises a decoder configured to generate said addresses into said microcode memory.

4. The apparatus according to claim 1, wherein predetermined sequences of said instruction codes of said first instruction set are used to address said microcode memory.

5. The apparatus according to claim 1, wherein addresses into said microcode memory are generated by a look-up-table in response to said instruction codes of said first instruction set.

6. The apparatus according to claim 1, wherein said instruction codes of said second instruction set comprise native instructions of a target processor.

7. The apparatus according to claim 6, wherein said target processor is selected from the group consisting of MIPS, ARM, and Motorola 68K.

8. The apparatus according to claim 3, wherein said microcode memory can be reprogrammed to support different processors.

9. The apparatus according to claim 1, wherein said circuit is configured to format the sequences of instruction codes of said second instruction set according to an opcode format of a processor.

10. The apparatus according to claim 1, wherein said circuit is configured to detect optimizable sequences of instruction codes on-the-fly.

11. The apparatus according to claim 1, wherein said circuit comprises a sequence optimization circuit.

12. The apparatus according to claim 1, wherein said circuit comprises a native instruction sequence generator circuit.

13. The apparatus according to claim 1, wherein said circuit is coupled between a processor and a memory system.

14. The apparatus according to claim 13, wherein said circuit is configured to (i) directly connect said processor and said memory system during a first state of operation and (ii) during a second state of operation, communicate with said processor as though said circuit was the memory system and communicate with said memory system as though said circuit was the processor.

15. The apparatus according to claim 1, wherein said instruction codes of said first instruction set comprise Java bytecodes.

16. The apparatus according to claim 1, wherein said circuit comprises a portion of a Java virtual machine implemented in hardware.

17. An apparatus comprising:

means for translating instruction codes of a first instruction set on-the-fly into addresses into a microcode memory containing sequences of instruction codes of a second instruction set that emulate a functionality of the instruction codes of said first instruction set;

means for receiving said instruction codes of said first instruction set; and

means for presenting said sequences of instruction codes of said second instruction set.

18. A method for on-the-fly translation of instructions of a first instruction set into instructions of a second instruction set comprising the steps of:

(A) receiving an instruction code of said first instruction set;

(B) generating an address into a microcode memory in response to said instruction code of said first instruction set using a hardware translator, wherein said address points to a sequence of instruction codes of said second instruction set that will emulate said instruction code of said first instruction set; and

(C) presenting said sequence of instruction codes of said second instruction set.

19. The method according to claim 18, wherein step B comprises the sub-step of:

selecting said address from a look-up table in response to said instruction code of said first instruction set.

20. The method according to claim 19, wherein step C further comprises the sub-step of:

optimizing said sequence of instruction codes of said second instruction set for a particular processor.

21. The apparatus according to claim 1, wherein said sequences of instruction codes of said second instruction set comprise one or more virtual stack references.

22. The apparatus according to claim 1, wherein said microcode memory further comprises one or more of (i) a size for each sequence of instruction codes of said second instruction set, (ii) a value representing how many bytes an instruction uses from said instruction codes of said first instruction set, and (iii) a stack change variable indicating whether the stack increases or decreases due to said instruction codes of said first instruction set and by how much.